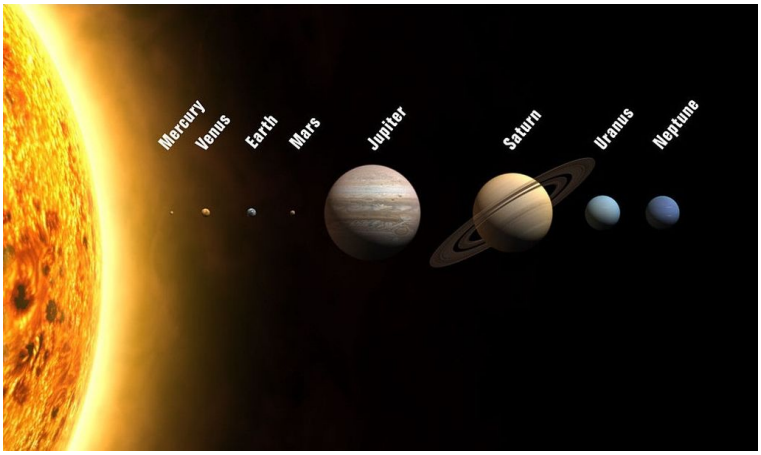
Solar system

Our **Solar System** comprises the Sun and objects that orbit it, whether they orbit it directly or by orbiting other objects that orbit it directly. Of those objects that orbit the Sun directly, the largest eight are the **planets** that form the **planetary system** around it. The remainder are significantly smaller objects, such as **dwarf planets** and small Solar System Bodies (SSSBs) such as **comets** and **asteroids**.



The Sun and the eight planets of the Solar System.

The eight planets of the Solar System can be aligned as follows according to increasing (average) distance to the Sun: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus en Neptune. When discovered in 1930, Pluto was initially considered to be the ninth planet of the Sun. This changed when it was downgraded to a dwarf planet in 2006 with the adoption of a formal definition of a planet. Eris, discovered in 2005, is the most massive known dwarf planet and and is at its most distant position roughly more than three times as far from the Sun as Pluto. Now, if we assign the values -8 up to and including 8 to the successive letters of the string JHMLCNVTURISEYAPO, we get:

```
      S+U+N = 3+0-3             = 0
M+E+R+C+U+R+Y = -6+4+1-4+0+1+5  = 1
    V+E+N+U+S = -2+4-3+0+3      = 2
    E+A+R+T+H = 4+6+1-1-7       = 3
      M+A+R+S = -6+6+1+3        = 4
J+U+P+I+T+E+R = -8+0+7+2-1+4+1  = 5
  S+A+T+U+R+N = 3+6-1+0+1-3     = 6
  U+R+A+N+U+S = 0+1+6-3+0+3     = 7
N+E+P+T+U+N+E = -3+4+7-1+0-3+4  = 8
    P+L+U+T+O = 7-5+0-1+8       = 9
      E+R+I+S = 4+1+2+3         = 10
```

Assignment

In this assignment, we call a string that is composed of an odd number of different letters a **letterstring**. All functions in this assignment take a letterstring as an argument, and must check that this string only contains letters, has an odd length and contains no duplicate letters (without making a distinction between uppercase and lowercase letters). If this is not the case, the functions must raise an `AssertionError` with the message `invalid letterstring`. Raising this `AssertionError` always takes precedence over any other exceptions that are explicitly raised by the functions.

In addition, one or more words can be passed to some functions. You may assume that these words only contain letters, without the need to check this explicitly. The **word value** of a word is an integer that is computed as the sum of the values of the individual letters. The **letter value** is determined using a lettterstring having length $2n + 1$ ($n \in \mathbb{N}$), whose successive letters are assigned the values

$-n$ up to and including n. No distinction should be made between uppercase and lowercase letters in determining the word value.

- Write a function lettervalue that takes a letterstring as an argument. For a given letterstring of length $2n + 1$ ($n \in \mathbb{N}$), the function must return a dictionary that maps the successive letters of the letterstring into the values $-n$ up to and including n. All keys of this dictionary should be uppercase letters.
- Write a function wordvalue that takes two arguments: a word and a letterstring. The function must return the word value of the given word, with the value of the individual letters based on the given letterstring. In case the given word contains letters that do not occur in the given letterstring (without making distinction between uppercase and lowercase letters), the function must raise an AssertionError with the message missing letters.
- Write a function alignment that takes two arguments: a **sequence** (a list or a tuple) of $m \in \mathbb{N}_0$ words and a letterstring. The function must return a Boolean value that indicates whether or not the words have successive word values $0, 1, \ldots, m - 1$. In determining the word values, the value of the individual letters should be based on the given letterstring.
- Write a function arrange1 that takes two arguments: a **list** of words and a letterstring. The function must sort the words in the given list according to increasing word value. In doing so, word values should be determined using the given letterstring. Words having the same word value must be sorted alphabetically, without making distinction between uppercase and lowercase letters.
- Write a function arrange2 that takes two arguments: a **sequence** (a list or a tuple) of words and a letterstring. The function must return a **tuple** containing the words from the given sequence sorted according to increasing word value. In doing so, word values should be determined using the given letterstring. Words having the same word value must be sorted alphabetically, without making distinction between uppercase and lowercase letters.

Make sure that the implementations of the function make optimal reuse of functionality that has already been implemented.

Example

```
>>> lettervalue('EARTH')
{'A': -1, 'H': 2, 'R': 0, 'E': -2, 'T': 1}
>>> lettervalue('Venus')
{'U': 1, 'S': 2, 'N': 0, 'E': -1, 'V': -2}
>>> lettervalue('Churyumov-Gerasimenk')
Traceback (most recent call last):
AssertionError: invalid letterstring

>>> wordvalue('SUN', 'JHMLCNVTURISEYAPO')
0
>>> wordvalue('mercury', 'JHMLCNVTURISEYAPO')
1
>>> wordvalue('Venus', 'JHMLCNVTURISEYAPO')
2
>>> wordvalue('EARTH', 'ABCDEFGHIJKLM')
Traceback (most recent call last):
AssertionError: missing letters

>>> alignment(['SUN', 'mercury', 'Venus'], 'JHMLCNVTURISEYAPO')
True
>>> alignment(['SUN', 'mercury', 'EARTH'], 'JHMLCNVTURISEYAPO')
False
>>> alignment(['Sun', 'mercury', 'Venus', 'EARTH', 'MARS', 'Jupiter', 'saturn', 'URANUS', 'Neptune', 'pluto'], 'JHMLCNVTURISEYAPO')
True

>>> planets = ['MARS', 'saturn', 'Jupiter', 'URANUS', 'Venus', 'mercury', 'EARTH', 'Sun', 'Neptune', 'pluto']
>>> arrange1(planeten, 'JHMLCNVTURISEYAPO')
>>> planets
```

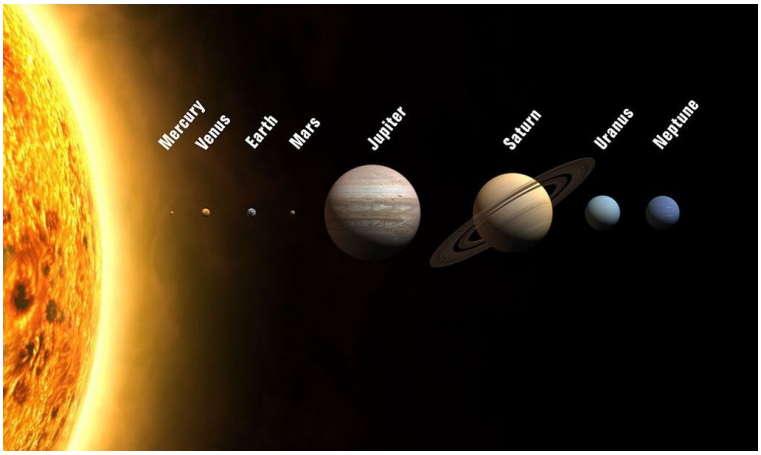
['Sun', 'mercury', 'Venus', 'EARTH', 'MARS', 'Jupiter', 'saturn', 'URANUS', 'Neptune', 'pluto']

```
>>> planets = ('MARS', 'saturn', 'Jupiter', 'URANUS', 'Venus', 'mercury', 'EARTH', 'Sun', 'Neptune', 'pluto')
>>> arrange2(planeten, 'JHMLCNVTURISEYAPO')
('Sun', 'mercury', 'Venus', 'EARTH', 'MARS', 'Jupiter', 'saturn', 'URANUS', 'Neptune', 'pluto')
```

Epilogue

Earth is the only planet not named after a god. If that's not mundane !

Ons **zonnestelsel** bestaat uit de Zon en alle objecten die er rond draaien, zijnde de objecten die er rechtstreeks rond draaien of die rond andere objecten draaien die zelf rond de Zon draaien. Van de objecten die rechtstreeks rond de zon draaien, zijn de acht grootste de **planeten** die samen het **planetenstelsel** van de Zon vormen. Alle andere objecten van het zonnestelsel zijn beduidend kleiner, zoals de **dwergplaneten** en de zogenaamde *small Solar System Bodies* (SSSBs) zoals **kometen** en **asteroïden**.



De Zon en de acht planeten in ons zonnestelsel.

De acht planeten van het zonnestelsel worden als volgt gerangschikt volgens oplopende (gemiddelde) afstand tot de Zon: Mercurius, Venus, Aarde, Mars, Jupiter, Saturnus, Uranus en Neptunus. Bij haar ontdekking in 1930 werd Pluto oorspronkelijk geclassificeerd als de negende planeet van de Zon. Dit veranderde echter in 2006 toen ze werd gedegradeerd tot dwergplaneet nadat een formele definitie voor planeten werd ingevoerd. Doorgaans staat Pluto verder van de zon dan Neptunus. Eris, ontdekt in 2005, is de zwaarste gekende dwergplaneet en staat op haar verste punt drie keer zo ver van de Zon als Pluto. Als we nu aan de opeenvolgende letters van de reeks JHMLCNVTURISEYAPO de waarden -8 tot en met 8 toekennen, dan krijgen we:

S+U+N	=	3+0-3	= 0
M+E+R+C+U+R+Y	=	-6+4+1-4+0+1+5	= 1
V+E+N+U+S	=	-2+4-3+0+3	= 2
E+A+R+T+H	=	4+6+1-1-7	= 3
M+A+R+S	=	-6+6+1+3	= 4

```
  J+U+P+I+T+E+R  =  -8+0+7+2-1+4+1  =  5
    S+A+T+U+R+N  =  3+6-1+0+1-3     =  6
    U+R+A+N+U+S  =  0+1+6-3+0+3     =  7
N+E+P+T+U+N+E+R  =  -3+4+7-1+0-3+4  =  8
      P+L+U+T+O  =  7-5+0-1+8       =  9
          E+R+I+S  =  4+1+2+3       =  10
```

Opgave

In deze opgave stellen we dat een **letterreeks** een string is die bestaat uit een oneven aantal verschillende letters. Aan alle functies uit deze opgave moet een letterreeks doorgegeven worden, waarbij de functies telkens moeten nagaan dat de gegeven string enkel bestaat uit letters, een oneven lengte heeft, en geen letters bevat die meermaals voorkomen (waarbij geen onderscheid gemaakt wordt tussen hoofdletters en kleine letters). Indien dat niet het geval is, dan moeten de functies een `AssertionError` opwerpen met de boodschap `ongeldige letterreeks`. Het opwerpen van deze `AssertionError` krijgt telkens voorrang op eventuele andere exceptions die expliciet door de functies opgeworpen worden.

Voorts worden aan sommige functies één of meer woorden doorgegeven, waarbij er mag van uitgegaan worden dat deze woorden enkel bestaan uit letters zonder dat dit expliciet moet gecontroleerd worden. De **woordwaarde** van een woord is een geheel getal dat bekomen wordt door de waarden van de individuele letters van het woord bij elkaar op te tellen. De **letterwaarde** wordt bepaald aan de hand van een letterreeks met lengte $2n + 1$ ($n \in \mathbb{N}$), waarvan de opeenvolgende letters de waarden $-n$ tot en met n toegewezen krijgen. Bij het bepalen van de woordwaarde wordt geen onderscheid gemaakt tussen hoofdletters en kleine letters.

- Schrijf een functie `letterwaarde` waaraan een letterreeks moet doorgegeven worden. Voor een gegeven letterreeks van lengte $2n + 1$ ($n \in \mathbb{N}$) moet de functie een dictionary teruggeven die de opeenvolgende letters van de letterreeks afbeeldt op de waarden $-n$ tot en met n. De sleutels van deze dictionary moeten allemaal hoofdletters zijn.
- Schrijf een functie `woordwaarde` waaraan twee argumenten moeten doorgegeven worden: een woord en een letterreeks. De functie moet de woordwaarde van het gegeven woord teruggeven, waarbij de waarde van de letters bepaald wordt op basis van de gegeven letterreeks. Indien het gegeven woord letters bevat die niet voorkomen in de gegeven letterreeks (zonder onderscheid te maken tussen hoofdletters en kleine letters), dan moet de functie een `AssertionError` opwerpen met de boodschap `ontbrekende letters`.
- Schrijf een functie `alignering` waaraan twee argumenten moeten doorgegeven worden: een **reeks** (een lijst of een tuple) van $m \in \mathbb{N}_0$ woorden en een letterreeks. De functie moet een Booleaanse waarde teruggeven die aangeeft of de woorden achtereenvolgens woordwaarde $0, 1, \ldots, m - 1$ hebben. Hierbij moet de woordwaarde bepaald worden aan de hand van de gegeven letterreeks.
- Schrijf een functie `rangschik1` waaraan twee argumenten moeten doorgegeven worden: een **lijst** van woorden en een letterreeks. De functie moet de woorden in de gegeven lijst volgens oplopende woordwaarde rangschikken. Hierbij moet de woordwaarde bepaald worden aan de hand van de gegeven letterreeks. Woorden met dezelfde woordwaarde moeten alfabetisch gerangschikt worden, zonder daarbij onderscheid te maken tussen hoofdletters en kleine letters.
- Schrijf een functie `rangschik2` waaraan twee argumenten moeten doorgegeven worden: een **reeks** (een lijst of een tuple) woorden en een letterreeks. De functie moet een **tuple** teruggeven waarin de woorden uit de gegeven reeks volgens oplopende woordwaarde gerangschikt zijn. Hierbij moet de woordwaarde bepaald worden aan de hand van de gegeven letterreeks. Woorden met dezelfde woordwaarde moeten alfabetisch gerangschikt worden, zonder daarbij onderscheid te maken tussen hoofdletters en kleine letters.

Zorg er bij de implementatie van bovenstaande functies voor dat er optimaal hergebruik gemaakt wordt van reeds geïmplementeerde functionaliteit.

Voorbeeld

```
>>> letterwaarde('EARTH')
{'A': -1, 'H': 2, 'R': 0, 'E': -2, 'T': 1}
>>> letterwaarde('Venus')
{'U': 1, 'S': 2, 'N': 0, 'E': -1, 'V': -2}
>>> letterwaarde('Churyumov-Gerasimenk')
Traceback (most recent call last):
AssertionError: ongeldige letterreeks

>>> woordwaarde('SUN', 'JHMLCNVTURISEYAPO')
0
>>> woordwaarde('mercury', 'JHMLCNVTURISEYAPO')
1
>>> woordwaarde('Venus', 'JHMLCNVTURISEYAPO')
2
>>> woordwaarde('EARTH', 'ABCDEFGHIJKLM')
Traceback (most recent call last):
AssertionError: ontbrekende letters

>>> alignering(['SUN', 'mercury', 'Venus'], 'JHMLCNVTURISEYAPO')
True
>>> alignering(['SUN', 'mercury', 'EARTH'], 'JHMLCNVTURISEYAPO')
False
>>> alignering(['Sun', 'mercury', 'Venus', 'EARTH', 'MARS', 'Jupiter', 'saturn', 'URANUS', 'Neptune', 'pluto'], 'JHMLCNVTURISEYAPO')
True

>>> planeten = ['MARS', 'saturn', 'Jupiter', 'URANUS', 'Venus', 'mercury', 'EARTH', 'Sun', 'Neptune', 'pluto']
>>> rangschik1(planeten, 'JHMLCNVTURISEYAPO')
>>> planeten
['Sun', 'mercury', 'Venus', 'EARTH', 'MARS', 'Jupiter', 'saturn', 'URANUS', 'Neptune', 'pluto']

>>> planeten = ('MARS', 'saturn', 'Jupiter', 'URANUS', 'Venus', 'mercury', 'EARTH', 'Sun', 'Neptune', 'pluto')
>>> rangschik2(planeten, 'JHMLCNVTURISEYAPO')
('Sun', 'mercury', 'Venus', 'EARTH', 'MARS', 'Jupiter', 'saturn', 'URANUS', 'Neptune', 'pluto')
```

Epiloog

De Aarde is de enige planeet die niet naar een god vernoemd werd. Als dat niet mondain is !